Microcontroller Technical Information

RX850 Pro V850 Real-Time OS Usage Restrictions	Document No.	ZMT-F35-10-0012		1/2
	Date issued	March 30, 2011		
	Issued by	MCU Tool Product Marketing Department MCU Software Division MCU Business Unit Renesas Electronics Corporation		
Related documents None	Notification classification	√	Usage restriction	
			Upgrade	
			Document modification	
			Other notification	

1. Affected product

   RX850 Pro Ver. 3.30 package (kernel RX850 Pro Ver. 3.21) and earlier versions

2. New restrictions

   The following restriction related to the kernel has been added:

   - No. 17 A cyclic handler might start up earlier than the specified cycle time.

3. Workarounds

   The following workaround is available for the above restriction. See attachment 1 for details.

   - No. 17 Do one of the following:
     - Do not allow an interrupt handler to issue the system call `act_cyc` for a cyclic handler when `TCYC_ON` has been specified.
     - Design the system so that only two cyclic handlers can be waiting to be started up at one time.

4. Modification schedule

   - No. 17 This will not be corrected, so regard it as a specification.

5. List of restrictions

   A list of restrictions in the RX850 Pro, including the revision history and detailed information, is described in the attachment.

6. Document revision history

RX850 Pro V850 Real-Time OS - Usage Restrictions

Document Number	Issued on	Description
SBG-T-2494	September 28, 2001	1st edition
SBG-TT-0007	November 2, 2001	Addition of new kernel restriction (No. 8)
SBG-TT-0096	May 9, 2002	Addition of new kernel restriction (No. 9)
ZBG-CD-05-0018	February 24, 2005	Addition of new kernel restriction (No. 10)
ZBG-CD-05-0027	March 31, 2005	Addition of new kernel restrictions (No. 11 and No. 12)
ZBG-CD-05-0035	May 27, 2005	Addition of new kernel restrictions (No. 13 and No. 14) Modification of figures in No. 11 in accordance with figures in No. 13
ZBG-CD-05-0057	June 13, 2005	Addition of description of kernel restriction (No. 13)
ZBG-CD-05-0068	July 20, 2005	Modification of description of kernel restriction (No. 11) Deletion of description of kernel restriction (No. 13)
ZBG-CD-06-0024	April 24, 2006	Modification of description of kernel restriction (No. 11) Addition of new kernel restrictions (No. 15 and No. 16)
ZBG-CD-09-0034	June 18, 2009	Addition of new CubeSuite-related restriction (No. 1)
ZMT-F35-10-0012	March 30, 2011	Addition of new kernel restriction (No. 17)

Restrictions Related to RX850 Pro Kernel

1. Product History

No.	Usage Restrictions	Kernel Version				
		3.11	3.13	3.15	3.20	3.21
1	Using `chg_icr()` and `ref_icr()` with V850E core	√	√	√	√	√
2	Size estimation for pool 0 in CF definition file	√				
3	`act_cyc` system call issuance	√				
4	Interrupt handler restoration	√	√			
5	Trace handler registration for AZ850	√	√			
6	Interrupt handler stack size specification during configuration	√	√			
7	System call `del_mpl`	√	√			
8	The program inadvertently loops at system call issuance when the RX850 Pro and user application are allocated to independent load modules.	√	√			
9	The AZ850 trace result is invalid if a static error is generated by a system call when using the AZ850 to analyze a module that uses the RX850 Pro.		√	√		
10	Interrupts cannot be executed after the `r19` register value is replaced by certain values in an initialization handler.	√	√	√		
11	The operation becomes undefined if a clock interrupt is acknowledged during cyclic handler processing.		√	√		
12	The operation becomes undefined if an attempt is made to wake up a task that does not exist when returning from an interrupt handler.	√	√	√		
13	–					
14	The operation becomes undefined when using system call `tget_blk`.	√	√	√		
15	If multiple clock interrupts are acknowledged, interrupts might no longer be acknowledged.				√	
16	Access using the `ep` register might be invalid depending on the option specified with a GHS compiler.	√	√	√	√	
17	A cyclic handler might start up earlier than the specified cycle time.	√	√	√	√	√

√: Applicable

2. Restriction Details

No. 1 Using `chg_icr()` and `ref_icr()` with V850E core

Description:

System calls `chg_icr()` and `ref_icr()` which access the interrupt control registers, might not work as expected when using the RX850 Pro on the V850E core.

Cause:

The RX850 Pro determines the address of an interrupt control register from the interrupt source number. However, the relationship between the interrupt control register addresses and interrupt source numbers in the V850E core is different from the rest of the V850 microcontroller, so the correct address cannot be obtained.

Workaround:

Control the interrupt control registers directly rather than using these system calls.

Correction:

This will not be corrected, so regard it as a specification.

No. 2 Size estimation for pool 0 in CF definition file

Description:

The interrupt factor conversion table will be damaged if memory pool 0 (SPOL0) is specified with a CF definition file as the exact size indicated in *Installation User's Manual*. Thus, an abnormality might occur in control of the interrupt handler. In addition, memory data might be damaged as a result of operation of an incorrect interrupt control register.

Cause:

This occurs because the cf850pro generates an invalid address as memory pool 0 used by the OS when specifying the size of memory pool 0 as the exact value based on the formula in *Installation User's Manual*. The RX850 Pro did not perform processing when the size of memory pool 0 excluding the resource management table was 0.

Workaround:

Specify the size of memory pool 0 to the exact size indicated plus 0x10 bytes.

Correction:

This issue has been corrected in Ver. 3.13.

No. 3 `act_cyc` system call issuance

Description:

If the `act_cyc` system call is issued from an interrupt immediately before execution of a cyclic handler, the activation of the cyclic handler is inadvertently cancelled.

Cause:

In the RX850 Pro, in the time between the elapse of the cyclic handler's activation time and when the cyclic handler is actually activated (while a timer handler is being executed), a flag is set in the cyclic handler's management structure indicating the ready state, and the cyclic handler is held waiting. In Ver. 3.11, if an interrupt is generated while this flag is set and the `act_cyc` system call is issued from the interrupt handler to the corresponding cyclic handler, this flag might be inadvertently cleared, causing the activation of the cyclic handler to be cancelled.

Workaround:

There is no workaround.

Correction:

This issue has been corrected in Ver. 3.13.

No. 4 Interrupt handler restoration

Description:

When `ret_wup` or equivalent processing is performed after an interrupt is generated and the interrupt handler operation is terminated, the `gp` and `ep` register values might not be restored. Note that this only occurs when the task ID woken up by `ret_wup` is invalid.

Cause:

Checking for errors in `ret_wup` is performed during task wake-up processing. If there is an error, the wake-up processing is skipped and resume processing from the interrupt handler is performed. However, if a task ID number range check error (`E_OACV`, `E_ID`) is detected, the processing sequence will change and the resume operation from the handler is performed without restoring the `gp` and `ep` register values.

Workaround:

- Disregard this error and execute the processing because the task ID number error is a static error.
- Do not use the `ret_wup` system call. Use the `wup_tsk` and `ret_wup` system calls in combination instead.

Correction:

This issue has been corrected in Ver. 3.15.

No. 5 Trace handler registration for AZ850

Description:

This restriction applies only when the RX850 Pro (`.system` section) and trace handler (`.text`) are allocated to independent load modules. In this case, even when the RX850 Pro calls the AZ850 trace handler registration routine, the `tp` register value is not switched. As a result, the trace handler is not registered correctly.

Cause:

The AZ850 trace handler registration routine is called by processing from the symbol `__az_defhdr` at the initialization of the RX850 Pro. This file exists in `trcent.o` (`.text` section) under `librx85p.a`. If this file is allocated in a load module that does not perform RX850 Pro's processing (in this case, if the file is allocated in the `.system` section by `start.o`, which initializes the RX850 Pro), the `tp` register is not switched when the operation jumps to `__az_defhdr`, causing an error.

Workaround:

An invalid address is called due to this behavior, causing a program loop at system startup (when the initial task starts up). Therefore, if the program loop does not occur, it can be judged that this restriction does not apply. When this behavior occurs, implement one of the following workarounds:

- Replace the AZ850 trace handler (`trcent.o`) with the corrected one (temporary workaround).
- Do not separate the RX850 Pro and trace handler into independent load modules.

Correction:

This issue has been corrected in Ver. 3.15.

No. 6 Interrupt handler stack size specification during configuration

Description:

The interrupt handler stack size must be specified in the configuration file, but even though 0x0 is specified, no error is output by the configurator (cf850pro).

Cause:

This is a judgment error inside the configurator (cf850pro).

From Ver. 3.15, the size can be specified in a range from 0x100 to 0x7ffffffc. If a value in a range from 0x0 to 0xff is specified, an error message `E2208: System stack size out of range` is output. For details about the interrupt stack capacity, see *5.4 CAPACITY OF STACK FOR INTERRUPT HANDLER* in *Installation User's Manual*.

Workaround:

Do not specify a value less than 0x100.

Correction:

This issue has been corrected in Ver. 3.15.

No. 7 System call `del_mpl`

Description:

When deleting a memory pool using the system call `del_mpl`, if the operation is performed in a state in which a memory block has been acquired from the target memory pool using the system call `get_blk`, `pget_blk`, or `tget_blk`, the memory area might not be released correctly.

If this occurs, the remaining capacity of the user memory is managed as being smaller than the actual capacity. If deletion and creation of a memory pool is repeatedly performed, therefore, no more memory pools can be created using the system call `cre_mpl`, causing an abnormal termination (`E_NOMEM`).

Cause:

When deleting a memory pool while a memory block is acquired, the acquired memory block should be released by the RX850 Pro. At this time, however, the memory block management size is not updated, causing this behavior.

Note, however, that this restriction does not apply when all the memory blocks are released by the system call `rel_blk` before a memory pool is deleted using the system call `del_mpl`.

Workaround:

Release all the memory blocks acquired from the target memory pool before deleting a memory pool.

Correction:

This issue has been corrected in Ver. 3.15.

No. 8 The program inadvertently loops at system call issuance when the RX850 Pro and user application are allocated to independent load modules.

Description:

The program might inadvertently loop at system call issuance if a user application (e.g. `.text`) is modified when the RX850 Pro and user application are allocated to independent load modules, or when the RX850 Pro is ROMized and its address is fixed.

This occurs when either of the following conditions (1) and (2) or (1) and (3) are satisfied at the same time.

(1) Renesas Electronics' compiler CA850 is used.

(2) The RX850 Pro (`.system`, `.system_int`, or `.system_cmn` section) and user application (e.g. `.text`) are allocated to independent load modules, and the `tp` (text pointer) values in the RX850 Pro and user application differ.

(3) The user application is modified after the RX850 Pro is ROMized and its address is fixed, and consequently, the `tp` value of the user application is changed.

Cause:

When a system call is issued, the AZ850 trace handler routine is executed. At this time, the location where the `tp` value of the RX850 Pro is used is found at the location where the `tp` value of the user application should be used. As a result, the program inadvertently loops because the correct address cannot be obtained from the system call table information at system call issuance when

- the RX850 Pro and user application are allocated to independent load modules, and the `tp` values in the RX850 Pro and user application differ

or

- the user application is modified after the RX850 Pro is ROMized and its address is fixed, and consequently the `tp` value of the user application is changed.

Workaround:

When this occurs, the program inadvertently loops when a system call is issued because an invalid address is called. In other words, if the system call is executed normally, it can be judged that this restriction does not apply. More precisely, the `tp` value code of the RX850 Pro (`.system`) and that of the user application (e.g. `.text`) are compared. If they are the same value, it can be judged that this restriction does not apply. Implement either of the following workarounds if this restriction applies.

- Replace the AZ850 trace handler (`trcent.o`) with the corrected one (temporary workaround).
- Do not separate the RX850 Pro and trace handler into independent load modules.
- Specify the previous `tp` value as `tp` and link it to the link directive file (address specification) when this behavior occurs due to program modification after ROMization of the RX850 Pro.

Correction:

This issue has been corrected in Ver. 3.15.

No. 9 The AZ850 trace result is invalid if a static error is generated by a system call when using the AZ850 to analyze a module that uses the RX850 Pro.

Description:

The AZ850 cannot create correct trace data if a static error that is detected inside the interface library preceding system call processing occurs as a result of system call issuance by the RX850 Pro.

The static error is one of the system call errors that does not occur during normal operation, and that has a return value of `E_ID` (specified ID number is invalid) or `E_PAR` (parameter specification is invalid), etc. This error indicates the return value marked with an asterisk (*) in the explanation of the return value under the explanation of the system call in the RX850 Pro Fundamental User's Manual.

Cause:

The AZ850 malfunctions because an invalid parameter is passed from the RX850 Pro to the AZ850 when a static error occurs.

Workaround:

The correct trace result can be obtained by clearing the static error of the system call before using the AZ850. If it is necessary to obtain the correct trace result even when a static error occurs, RX850 Pro Ver. 3.16, in which this restriction is corrected, must be used. In such a case, Renesas Electronics will provide separate support as a temporary measure.

Correction:

This issue has been corrected in Ver. 3.20.

No. 10 Interrupts cannot be executed after the r19 register values are replaced by certain values in an initialization handler.

Description:

- When an initialization handler is complete, the values of the r19 register are replaced by the following values:

  xxxx xxxx xxxx xxxx xxxx xxxx 010*x xxxx* (binary data, *x* is any value)

- A maskable interrupt occurs in the period from the completion of the initialization handler to the next task activation.

When the above conditions are satisfied, maskable interrupts whose priority is lower than that of the generated maskable interrupt can no longer be executed.

With the RX850 Pro, two nucleus-common object, rxcore.o and rxtmcore.o, are provided. When using rxtmcore.o, this restriction applies to all maskable interrupts. When using rxcore.o, this restriction applies to maskable interrupts except for the one registered as a timer using the configurator.

Cause:

The RX850 Pro uses the r19 register for PSW register backup at initialization processing. The r19 register values are written back to the PSW register after the initialization handler is completed. The behavior occurs if the PSW register values are changed to NP = 0, EP = 1, and ID = 0 (values shown below).

xxxx xxxx xxxx xxxx xxxx xxxx 010*x xxxx* (binary data, *x* is any value)

If the above values are written to the PSW register and then an interrupt is executed, EP remains 1 when the program is restored from the interrupt (reti). At this time, the flag of ISPR remains 1. As a result, maskable interrupts whose priority is lower than that of the generated maskable interrupt can no longer be executed.

Workaround:

Save and restore the r19 register values within the initialization handler.

Example:

```
void varfunc()
{
    …  /* local variable declaration */
    _asm("add   -4, sp");
    _asm("st.w  r19, 0[sp]");
    …  /* function processing */
    _asm("ld.w  0[sp], r19");
    _asm("add   4, sp");
    return;
}
```

Correction:

This issue has been corrected in Ver. 3.20.

No. 11 The operation becomes undefined if a clock interrupt is acknowledged during cyclic handler processing.

Description:

If a clock interrupt is acknowledged during cyclic handler processing, the PSW register values are corrupted by invalid values and the subsequent operations become undefined.

This occurs when `rxcore.o` is used as a nucleus-common object but does not occur in cases where `rxtmcore.o` is used.

- Example of undefined operation 1:

  If the ID bit of the corrupted PSW register is set to 1, then the period during which interrupts are disabled during clock interrupt processing is extended for approximately ten instructions. Even if this behavior occurs and the interrupt disabled period is extended, the period does not exceed the longest OS interrupt disabled period.

  The PSW register restores the normal state if the OS performs processing to return from the interrupt (by issuing the `reti` instruction).

- Example of undefined operation 2:

  If the EP bit is illegally set to 1, the ID bit to 0, and the NP bit to 0 in the PSW register, and if a clock interrupt is acknowledged multiple times at a point where interrupts are enabled in the OS, immediately after the illegal value setting, the ISPR register values becomes invalid. (Even after the clock interrupt is completed, the bit that indicates the clock interrupt priority remains set to 1). As a result, interrupts with a priority lower than that of the clock interrupt will no longer be acknowledged.

Cause:

The PSW register values are corrupted because the execution returns from the interrupt without saving the PSW register values in the stack, and undefined values are set to the PSW register. However, the stack pointer operates normally even if this behavior occurs.

Workaround:

Change the nucleus-common object from `rxcore.o` to `rxtmcore.o`.

Correction:

This issue has been corrected in Ver. 3.20.

No. 12 The operation becomes undefined if an attempt is made to wake up a task that does not exist when returning from an interrupt handler.

Description:

The ID of a task to be woken up can be specified as the return value of an interrupt handler. If `TSK_NULL` (= 0) or a task ID that does not exist is specified, the execution returns from the interrupt handler without performing task wake-up processing.

If a signed negative number (0x80000000 to 0xffffffff) is specified as the ID for a task that does not exist, the subsequent operation becomes undefined.

Cause:

If a task ID of a signed negative number is specified when returning from an interrupt handler, the validity of the task ID is not confirmed. Consequently, the program accesses an invalid address and the subsequent operation becomes undefined.

Workaround:

Do not specify a task ID with signed negative numbers as the return value for an interrupt handler.

Correction:

This issue has been corrected in Ver. 3.20.

No. 13 <Deleted>

No. 14 The operation becomes undefined when using system call `tget_blk`.

Description:

If all of the following conditions are satisfied, the subsequent operations become undefined.

(1) A task, which has been waiting for memory block acquisition with system call `tget_blk`, times out.
(2) In addition to the task that times out in (1), there is another task waiting for memory block acquisition.
(3) The task that times out in (1) is the first task in the memory block queue.
(4) Another task or a cyclic handler times out at the same time as (1).

Cause:

When all of the above conditions are satisfied, the contents of the register that holds address values are corrupted in RX850 Pro internal processing. As a result, branch operations that use these address values become undefined.

Workaround:

There is no workaround.

Correction:

This issue has been corrected in Ver. 3.20.

No. 15 If multiple clock interrupts are acknowledged, interrupts might no longer be acknowledged.

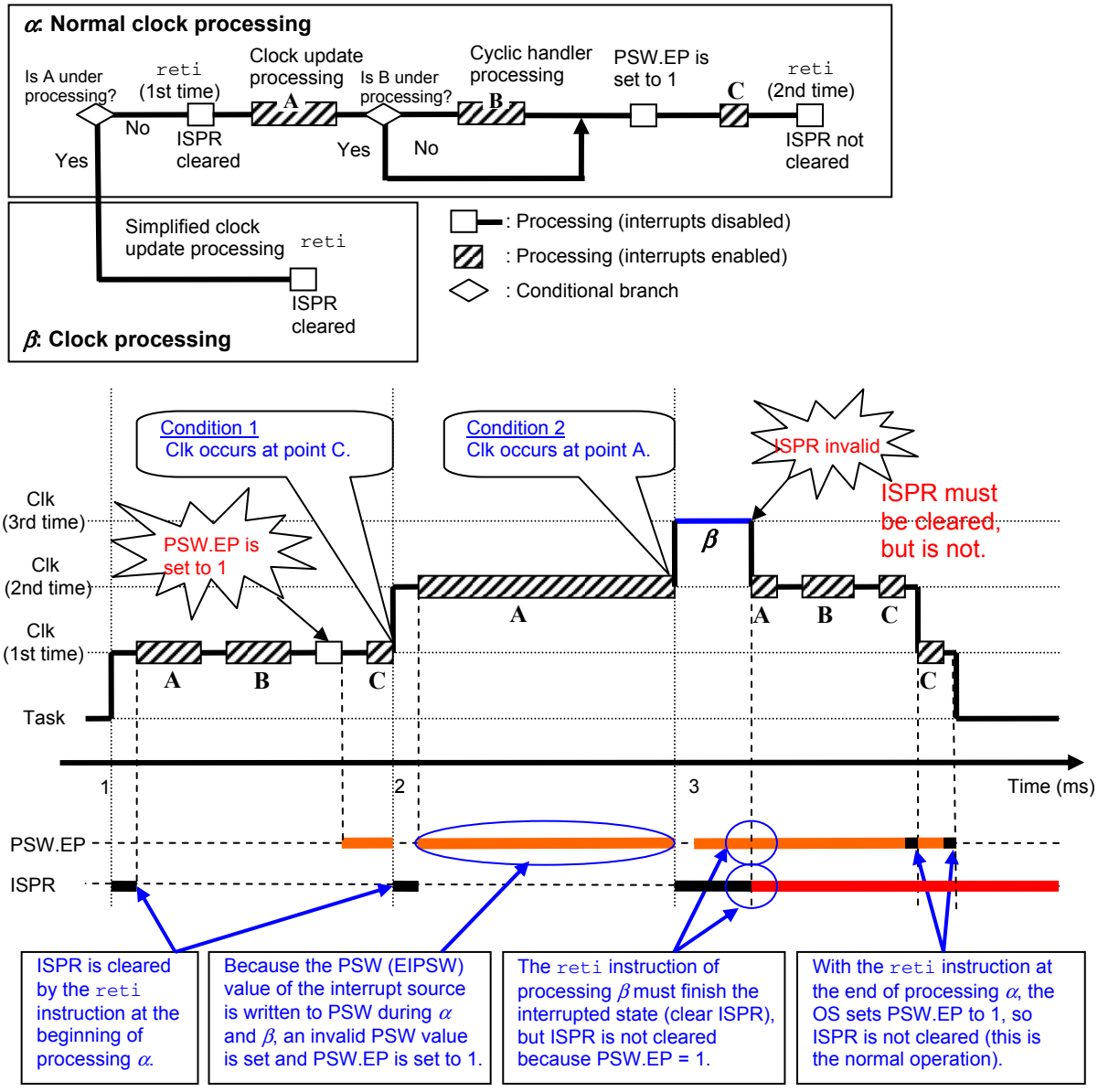
Description:

If multiple clock interrupts are acknowledged, the priority bit of the ISPR register corresponding to the clock interrupt remains set to 1. As a result, interrupts with a priority lower than that of the clock interrupt will no longer be acknowledged.

This occurs when `rxcore.o` is used as a nucleus-common object but does not occur in cases where `rxtmcore.o` is used.

Cause:

If an interrupt occurs at a specific timing during clock interrupt processing (point C in the figure below), EP remains set to 1 when the execution returned from the interrupt (by `reti`) and the ISPR register flag remains set to 1. As a result, interrupts with a priority lower than that of the acknowledged interrupt will no longer be acknowledged. The following summarizes the clock interrupt processing.


α: Normal clock processing
Is A under processing?
reti (1st time)
Clock update processing
A
Is B under processing?
B
Cyclic handler processing
PSW.EP is set to 1
C
reti (2nd time)
No
ISPR cleared
Yes
Yes
No
ISPR not cleared
Simplified clock update processing
reti
ISPR cleared
β: Clock processing
: Processing (interrupts disabled)
: Processing (interrupts enabled)
: Conditional branch
Condition 1
Clk occurs at point C.
Condition 2
Clk occurs at point A.
ISPR invalid
ISPR must be cleared, but is not.
Clk (3rd time)
Clk (2nd time)
Clk (1st time)
PSW.EP is set to 1
β
A
A B C
A B C
C
Task
1
2
3
Time (ms)
PSW.EP
ISPR
ISPR is cleared by the reti instruction at the beginning of processing α.
Because the PSW (EIPSW) value of the interrupt source is written to PSW during α and β, an invalid PSW value is set and PSW.EP is set to 1.
The reti instruction of processing β must finish the interrupted state (clear ISPR), but ISPR is not cleared because PSW.EP = 1.
With the reti instruction at the end of processing α, the OS sets PSW.EP to 1, so ISPR is not cleared (this is the normal operation).

Workaround:

Change the nucleus-common object from `rxcore.o` to `rxtmcore.o`.

Correction:

This issue has been corrected in Ver. 3.21.

No. 16 Access using the ep register might be invalid depending on the option specified with a GHS compiler.

Description:

If an application is compiled using a GHS compiler with any of the following optimization options specified, a code that uses the ep register as a temporary register might be output.

- -OS
- -notda
- -allocate_ep (option added in MULTI Ver. 4.0.2 Rel. 7.0.1 and later)

If an interrupt (directly/indirectly activated interrupt) is generated by the RX850 Pro in a section in which the ep register is used as the temporary register (the ep register values have been overwritten), the subsequent operations might be undefined.

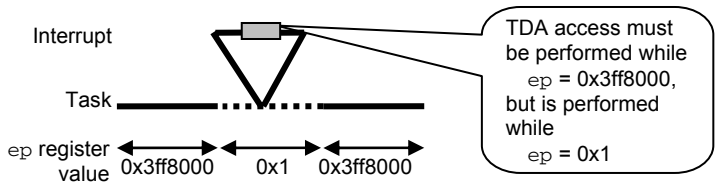
Cause:

This occurs because handling of the ep register differs between the RX850 Pro and GHS compiler.

The GHS compiler handles the ep register as a base register for accessing TDA, and also handles the ep register as a temporary register. In contract, the RX850 Pro only handles the ep register as a base register for accessing TDA. Due to this difference, the problems described in the following examples occur if the RX850 Pro operates while the GHS compiler is using the ep register as a temporary register.

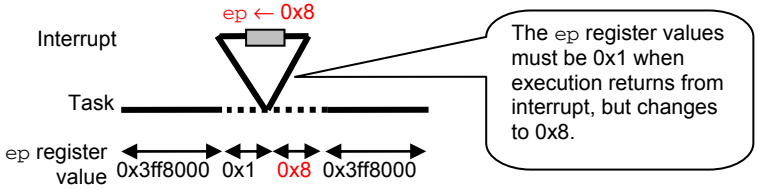
Example 1

If an interrupt is generated by RX850 Pro in a section in which the ep register has been overwritten (dotted line in the figure below) and TDA is accessed by the interrupt routine (shaded portion in the figure below), the execution will access an invalid address. This is because the interrupt routine activated by the RX850 Pro cannot recognize that the ep register is used as the temporary register, and it cannot recognize that the interrupt handler accesses TDA.



Example 2

If an interrupt is generated in a section in which the ep register has been overwritten (dotted line in the figure below) and the ep register is overwritten by the interrupt routine (shaded portion in the figure below), the temporary register values become invalid when the execution returns from the interrupt and the operation becomes undefined. This is because the interrupt routine activated by the RX850 Pro does not save and restore the ep register values.



Workaround:

Specify the options by using the GHS compiler as follows, so as not to use the `ep` register as the temporary register.

- When specifying `-OS` or `-notda`, also specify `-Z1412` together.
- Do not specify `-allocate_ep`.

`-Z1412` is an internal option, and is only available in currently available compiler Ver. 4.0.7 Rel. 7.0.3 or earlier. The specification might change in conjunction with the compiler revision, so consult a GHS compiler dealer for whether this restriction applies to compilers released after April 2006.

Correction:

This issue has been corrected in Ver. 3.21.

No. 17 A cyclic handler might start up earlier than the specified cycle time

Description:

When an interrupt handler issues the system call `act_cyc` for a cyclic handler and `TCYC_ON` has been specified, the cyclic handler might start up earlier than the specified cycle time.

This operation occurs if all the following conditions are satisfied:

- Cyclic handler A starts after its cycle phase elapses following the issuance of the timer interrupt before timer interrupt T (timer interrupt T-1).
- At least two cyclic handlers, excluding cyclic handler A that was started by timer interrupt (T-1), are waiting to start up when timer interrupt T occurs.
- When timer interrupt T occurs, there is a cyclic handler whose cycle time exceeds the time at which cyclic handler A will start up next time.
- When timer interrupt T occurs, another interrupt occurs during the processing to compare the cycle time of cyclic handler A and the other cyclic handlers waiting to start up to determine the handler startup order, and `act_cyc` is issued for the relevant cyclic handler with `TCYC_ON` having been specified in that interrupt handler.

When this operation occurs, among the cyclic handlers waiting to start up when timer interrupt T occurs, all the handlers whose cycle time is longer than that of cyclic handler A start up earlier than the specified time. The time until cyclic handler A starts up next time is then brought forward.

Workaround:

Do one of the following:

- Do not allow an interrupt handler to issue the system call `act_cyc` for a cyclic handler when `TCYC_ON` has been specified.
- Design the system so that only two cyclic handlers can be waiting to be started up at one time.

Correction:

This will not be corrected, so regard it as a specification.

CubeSuite-Related Usage Restrictions

1. Product History

No.	Usage Restrictions	Package Version	
		3.21 or earlier	3.30
1	Restriction that the **Help** menu cannot be opened from a message by using the **F1** key		√

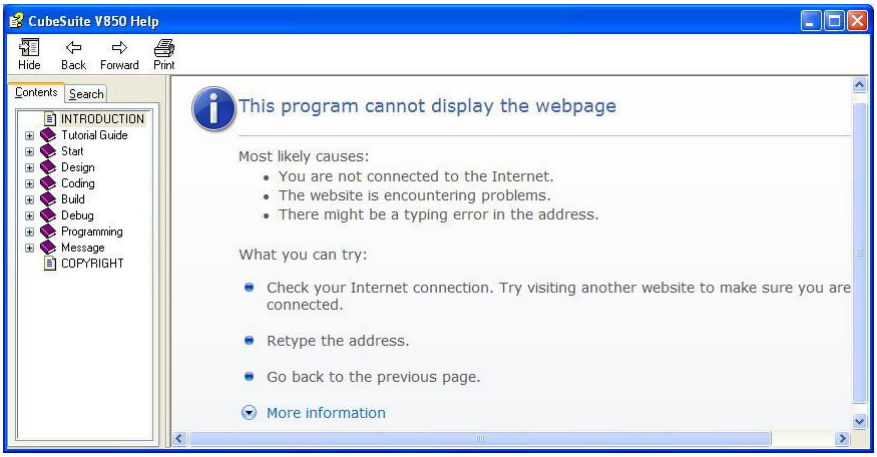
√: Applicable

2. Restriction Details

No. 1 Restriction that the **Help** menu cannot be opened from a message by using the **F1** key

Description:

If the **F1** key is pressed while the cursor is pointing to a message output by the build tool in an RX850 Pro project, the **Help** menu does not open with an appropriate explanation. Instead, the following **Help** message is displayed:



Workaround:

On the menu bar, click **Help** and then **Help for CubeSuite** to open the **CubeSuite Help** dialog box. Search for the appropriate **Help** message in this dialog box.

Correction:

This issue has been corrected in the Realtime OS Build Tool Plug-in (Common) Ver. 1.01.